Chapter 8: The IO Library

8.3 `string` Stream

The `sstream` header defines three types to support in-memory IO; these types read from or write to a `string` as if the `string` were **an IO stream**.

The `istringstream` type reads a `string`, `ostringstream` writes a `string`. Table below lists specific operations (in addition to standard IO stream operations) for them.
文字

sstream `strm;`	`strm` is an unbound `stringstream`. *sstream* is one of the types defined in the `sstream` header.
sstream `strm(s);`	`strm` is an *sstream* that holds a copy of the `string` `s`. This constructor is `explicit` (§ 7.5.4, p. 296).
`strm.str()`	Returns a copy of the `string` that `strm` holds.
`strm.str(s)`	Copies the `string` `s` into `strm`. Returns `void`.

8.3.1. Using an `istringstream`

The `istringstream` type reads a `string`. The characters in the string sequence can be extracted from the `istringstream` object using any operation allowed on input streams (e.g., >>). An `istringstream` is often used when we NEED to do some work with **individual words within a string**.

<u>Example:</u> A phone dataset typically consists of the name of the person and his/her phone numbers. A person can have many phone numbers associated with him. For example, phone number from work, from home and mobile. Typical records in an input file (`input.txt`) might look like:

```
John 33664275 0937495295
Lutz 33664175 0912120212 0987346123
```

Each record in this file starts with a name, which is followed by one or more phone numbers. We'll start by defining a simple class to represent our input data:

```
struct PersonInfo {
    string name;
    vector<string> phones;
};
```

Our program will read the data file and build up a `vector` of `PersonInfo`. We'll extract the name and phone numbers for each line (work to do with **individual words within a line**):

declare first, use later
只叫你宣告之後 就可以使用

interface || implementation
把 ___.h include進來

一般來說 .h
只會做 宣告
(簡單一兩行的implementation)

不會做 定義

宣告放在 class.h
- class的定義
- non member function的宣告
- 不可使用 using namespace

定義放在 class.cpp
- 可以 using namespace
- 要 include "class.h"
- 定義 member function … 記得 class scope

Phone.cpp

```
#include <iostream>
#include <sstream>
#include <fstream>
#include <vector>
#include <string>
#include <cstdlib>
using namespace std;

// members are public by default
struct PersonInfo {
    string name;
    vector<string> phones;
};

ifstream& open_file(ifstream& in, const string& file)
{
    in.close();     // close in case it was already open
    in.clear();     // clear any existing errors
    // if the open fails, the stream will be in an invalid state
    in.open(file.c_str()); // open the file we were given
    return in; // condition state is good if open succeeded
}

void printRecords(ostream &os, const vector<PersonInfo>& people)
{
    for (const auto &entry : people) {    // for each entry in people
        os << entry.name << " has " << entry.phones.size()
            << " phones and the numbers are: ";
        for (const auto &nums : entry.phones) {  // for each number
            os << nums << " ";
        }
        os << endl;
    }
}

int main()
{
    ifstream fin;
    string file_name;
    cout << "Enter the file name: ";
    cin >> file_name;
    if (!open_file(fin, file_name)) {
        cerr << "Complain: I cannot find the file" << endl << endl;
        cerr << system("dir") << endl; //only for windows OS
        return -1;
    }

    string line, word;
```

```
    vector<PersonInfo> people;

    // read the input a line at a time until end-of-file (or other error)
    while (getline(fin, line)) {
        PersonInfo info;           // object to hold this record's data
        istringstream record(line); // bind record to line we just read
        record >> info.name;        // read the name
        while (record >> word)      // read the phone numbers
            info.phones.push_back(word);  // and store them
        people.push_back(info); // append this record to people
    }

    printRecords(cout, people);
    return 0;
}
```

```
Enter the file name: input.txt
John has 2 phones and the numbers are: 33664275 0937495295
Lutz has 3 phones and the numbers are: 33664175 0912120212 0987346123
```

Example: Another common usage of `istringstream` is to **retrieve the numeric value** from the string. Reading an `istringstream` automatically converts from the character representation of a numeric value to its corresponding arithmetic value:

CovertNum.cpp

```
#include <iostream>
#include <sstream>
#include <string>
using namespace std;

int main()
{
    string s = "Some numerical values are 23 5 7";
    cout << s << endl;
    istringstream input_istring(s);
    string dump;
    int val1, val2, val3;
    input_istring >> dump >> dump >> dump >> dump
        >> val1 >> val2 >> val3;
    cout << "The sum is: " << val1 + val2 + val3 << endl;
    return 0;
}
```

```
Some numerical values are 23 5 7
The sum is: 35
```

Ex81.cpp

In-class Exercise 8.1: Input two stocks with each stock containing stock name, number of

share owned and the share price and report the most valuable one. For example:

```
Enter the first share (name, num_holding, share_price): AAPL 1 100.73
Enter the second share (name, num_holding, share_price): MSFT 2 44.03
AAPL with total value of 100.73 is more valuable.
```

The client code looks like:

StockClient.cpp

```
#include <iostream>
#include "Stock.h"

using namespace std;

int main()
{
   cout << "Enter the first share (name, num_holding, share_price): ";
   string line;
   getline(cin, line);
   Stock s0(line);

   cout << "Enter the second share (name, num_holding, share_price): ";
   getline(cin, line);
   Stock s1(line);
   Stock top = s0.topval(s1);
   cout << top.getName() << " with total value of "
      << top.getTotalVal() << " is more valuable." << endl;
   return 0;
}
```

And `Stock.h` looks like:

Stock.h

```
#ifndef STOCK_H
#define STOCK_H
#include <string>

class Stock
{
public:
   Stock(std::string record);
   const Stock& topval(const Stock& s) const;
   std::string getName() const {return name;}
   double getTotalVal() const {return total_val;}
private:
   std::string name;
   unsigned share_num;
   double share_val;
   double total_val;
};

#endif // STOCK_H
```

Implement the `Stock.cpp`.

A:

8.3.2. Using an `ostringstream`

The `ostringstream` type writes a `string` and characters can be inserted into the `ostringstream` object with any operation allowed on output streams (e.g., <<). With `ostringstream`, we can easily build up our output a little at a time and finally output the combining string.

CombineNumeric.cpp

Example

```
#include <string>
#include <iostream>
#include <sstream>
using namespace std;

int main() {
    cout << "Enter an English expression for an integer: ";
    string s;
    getline(cin, s);
    cout << "Enter an integer: ";
    int num;
    cin >> num;
    ostringstream oss;
    oss << s << ": " << num;
    cout << oss.str() << endl;
    return 0;
}
```

```
Enter an English expression for an integer: One hundred and one
Enter an integer: 101
One hundred and one: 101
```